MColl: MONTE Collocation Trajectory Design Tool

Daniel Grebow and Thomas Pavlak

Jet Propulsion Laboratory
California Institute of Technology

Presentation Outline

1. MColl: What is it?

2. Overview of collocation and mesh refinement methods

3. Implementation in MColl and current user-capability

4. Example problems

MColl (MONTE Collocation)

Prototype software resulting from three-year R&D effort funded by NASA's Advanced Multi-Mission Operations System (AMMOS)

Goal: Enable rudimentary, low-thrust trajectory optimization in MONTE

MONTE: JPL's Mission-design and Operations Navigation Toolkit Environment

Collocation: Numerical method for computing solutions to differential equations

Collocation Overview

- Collocation starts with a discretized trajectory approximation defining nodes of polynomials
- The collocation problem is 'solved' when the time derivatives of the polynomials match the dynamical differential equations at every node
- Equivalent to implicit Runge-Kutta integration schemes (Weiss, 1974)
- Different strategies:
    - Polynomial bases (Lagrange interpolation, B-splines, Chebyshev polynomials, etc.)
    - Degree of polynomial
    - Number of polynomial segments (one, a few, or many)
    - Placement of nodes (LGL, LGR(r), LG, CGL, etc.)
- Historically used to solve BVPs (e.g., COV indirect method)
    - In astrodynamics community today, more commonly used in direct transcription
        - NLP that results from collocation is solved by some 3rd-party sparse optimization software
- Collocation software: COLSYS, COLDAE, AUTO, OTIS, SOCS, DIDO, DIRCOL, PROPT, GPOPS-II

Mesh Refinement

- After the collocation problem is solved, compute error for each segment
- Segment boundary times or polynomial degree adjusted to meet error tolerance
- Mesh refinement is equivalent in principle to adaptive step-size for explicit integration schemes
- Mesh refinement is essential for computing an accurate solution
- Strategies:
  - Adjust degree of polynomial
  - Equidistribute error
    - Compare to higher order solution, such as n^{th} + 1 degree
    - Sundman transformation
    - n^{th}-derivative differencing scheme (de Boor, 1973)
  - Change number of segments
    - Once error is equally distributed, simple equation estimates number of segments needed to meet error tolerance
    - Use 3rd-party explicit propagation software to check error (CEP)

MColl Implementation & Capability

- Written in Python
- Odd degree polynomials (*n*=3,5,7,9), LGL points
  - Polynomials satisfy state and mass continuity at segment boundaries
- Spacecraft accelerations
  - MONTE models: point mass or full gravity, SRP (solar sailing), drag, etc.
  - High-fidelity SEP thruster, solar array, and bus power models
- Boundary constraints, path constraints, point constraints, and objective easy to specify
  - Constraints and objective can be any MONTE computable quantity, or computed by user
- User can specify multiple legs with different parameters for optimization
- Derivatives computed on per-segment basis using MONTE's built-in automatic differentiation capability
- NLP solved with sparse minimum-norm solution, or for direct optimization: IPOPT, KNITRO
- Mesh refinement algorithms available: de Boor, CEP, or hybrid
- Final solution validated by propagating with JPL's DIVA explicit integrator

SEP Throttling in MColl

- Segment start mass $m_{0,i}$ end mass $m_{f,i}$ are NLP variables
- For each segment, a 'throttling' inequality constraint function s_i is enforced such that

$$0 \le s_i \le 1$$

where

$$s_i = \frac{m_{0,i} - m_{f,i}}{\dot{m}_{\max,i} \Delta t_i}$$

Then the segment thrust T_i and mass flow $\dot{m}_i$ are

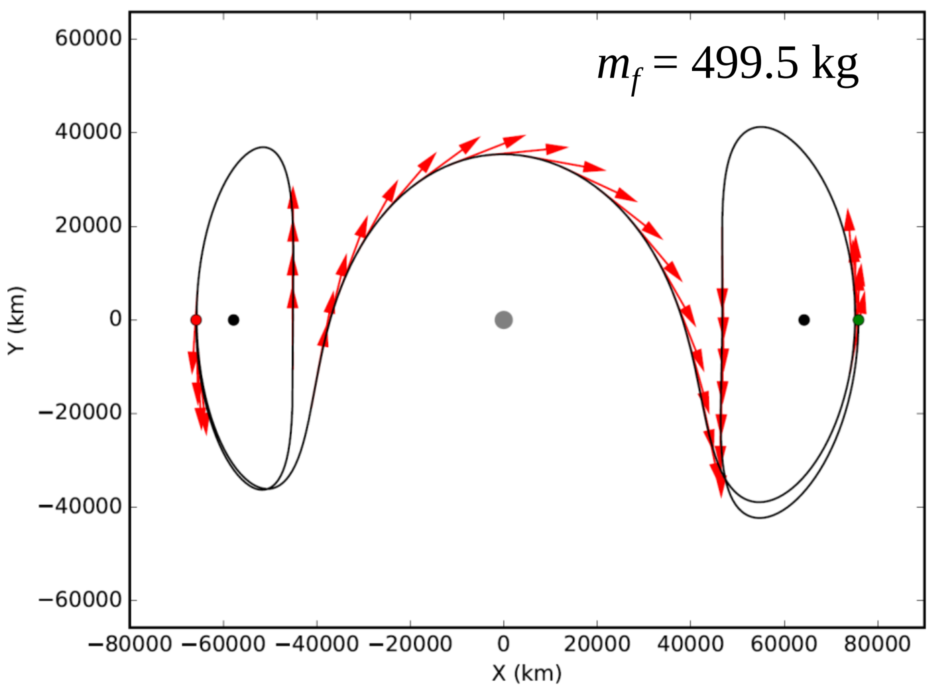
$$T_i = s_i T_{\max,i}, \quad \dot{m}_i = s_i \dot{m}_{\max,i}$$

$T_{\max,i}$ and $\dot{m}_{\max,i}$ are polynomial functions of input power

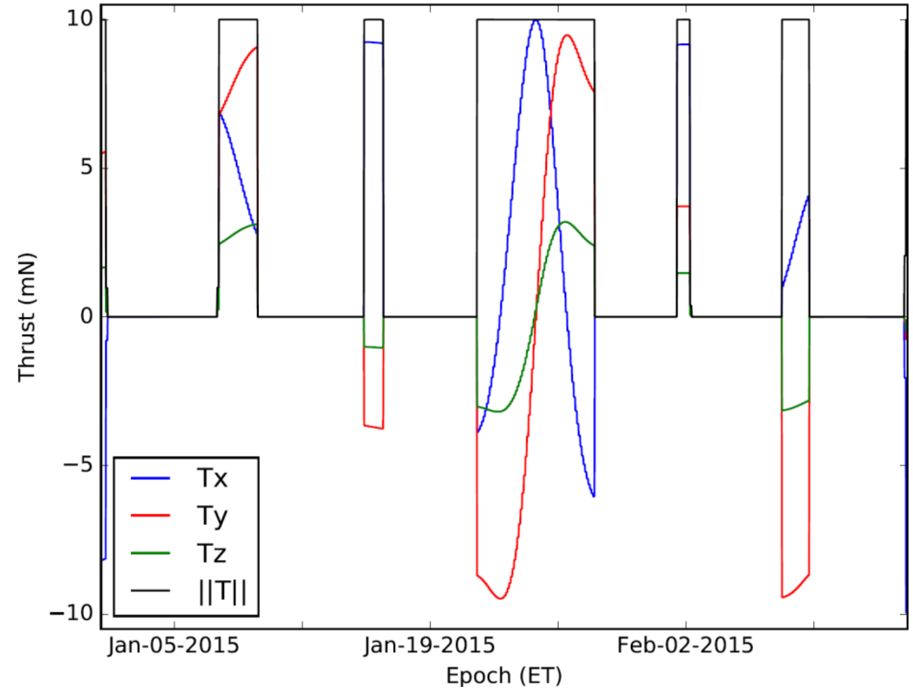
Lyapunov-to-Lyapunov Low-Thrust Transfer

Jet Propulsion Laboratory
California Institute of Technology

- Initial spacecraft mass: 500 kg
- Thrust parameters: constant thrust 10 mN, Isp = 2,000 sec
- Objective: maximize final mass
- Compute time: 21 seconds

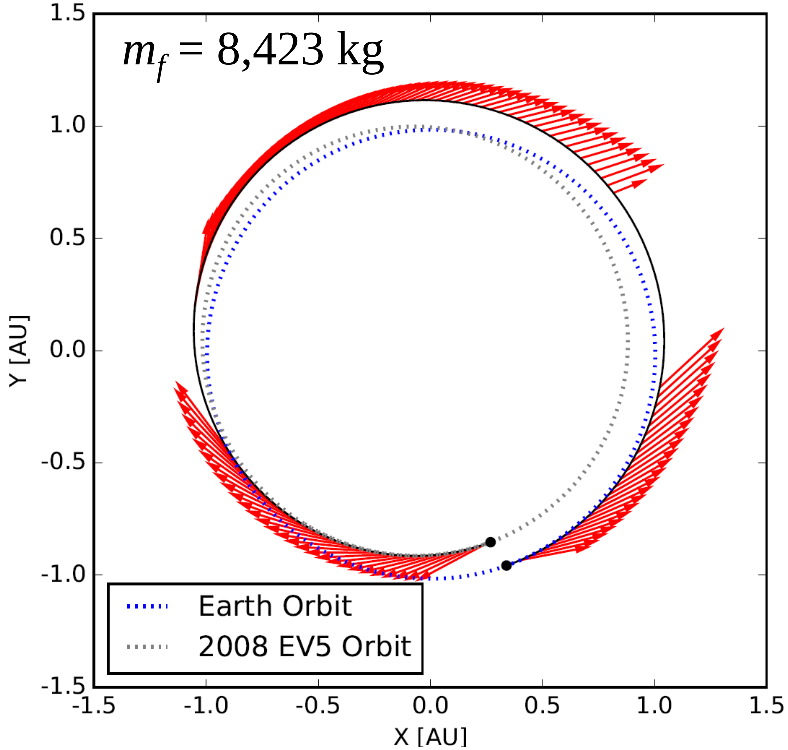


Optimal thrust profile

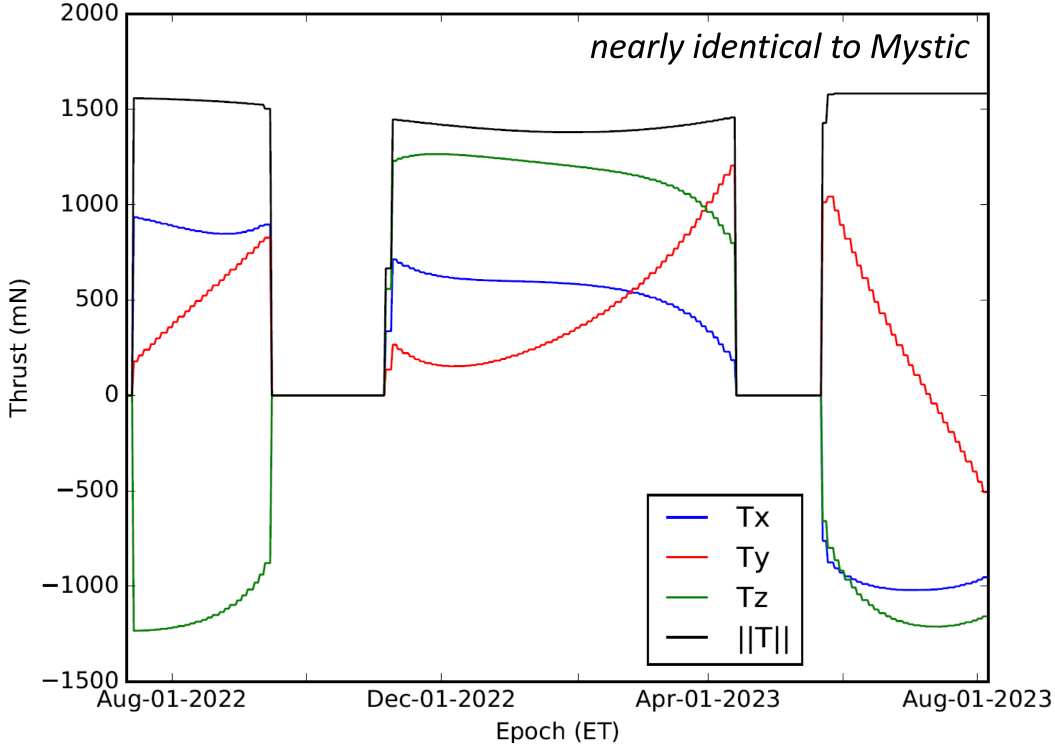


Earth-to-Asteroid Rendezvous (ARRM)

- Initial spacecraft mass: 9,945 kg
- Thrust parameters: 3 HERMeS high-efficiency thrusters, 90% duty cycle, $1/r^2$ array with P_0 = 47 kW, bus power 500W
- Objective: maximize final mass
- Compute time: 25 seconds (same in Mystic)

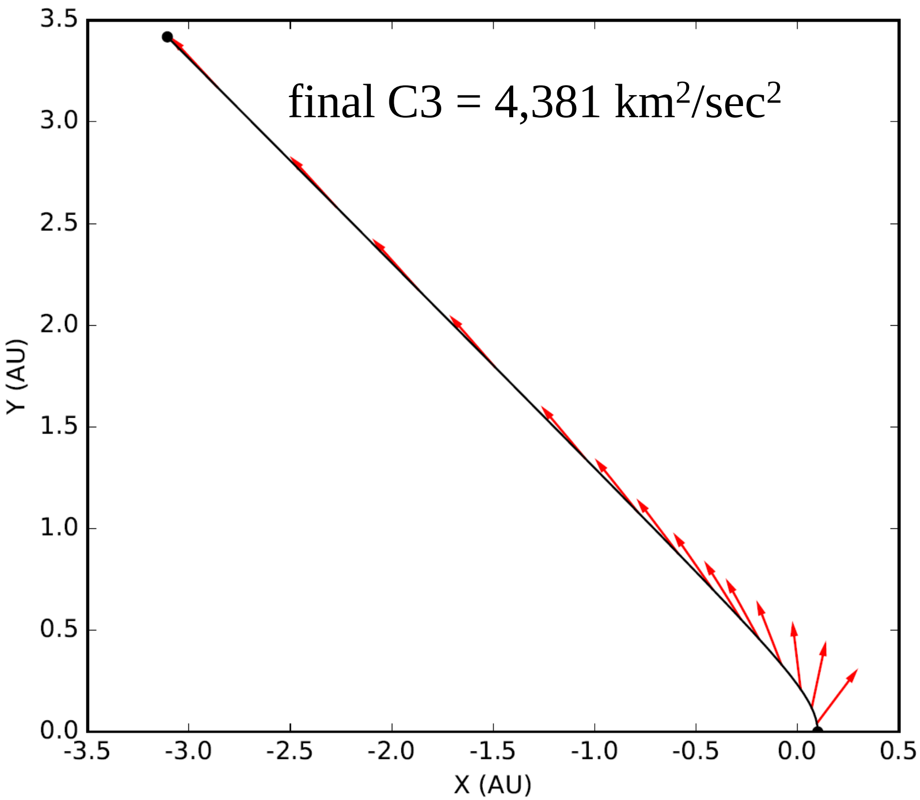


Optimal thrust profile

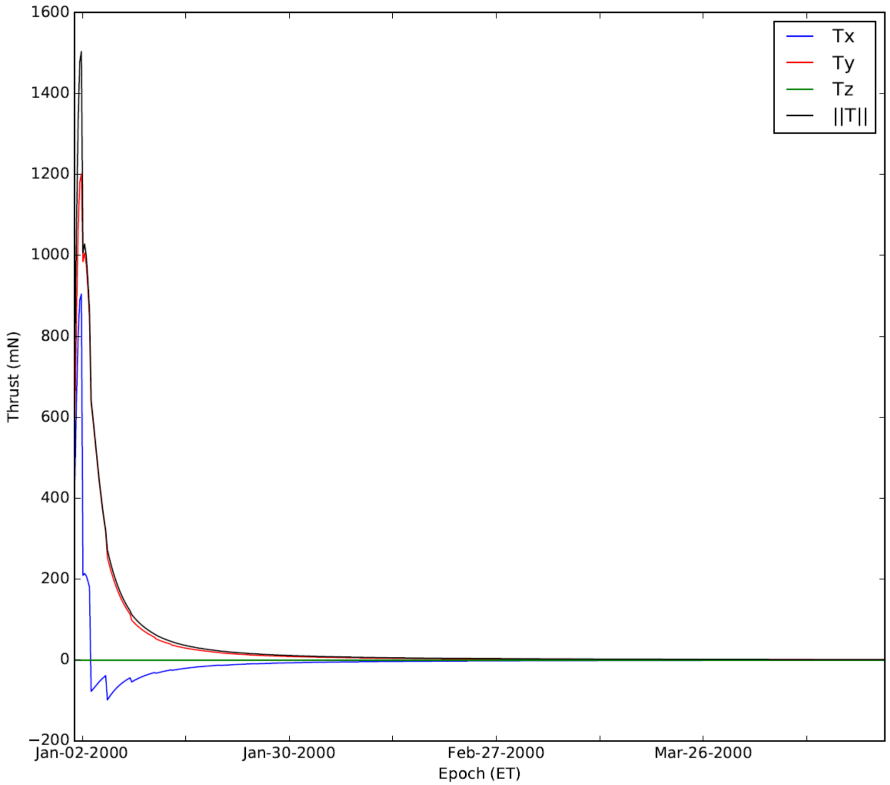


Solar Sail Solar System Escape

- Initial state: 0.1 AU perihelion parabola, mass = 20 kg
- Thrust parameters: ideal sail, 50x50m^2
- Objective: maximize final C3
- Compute time: 82 seconds (MALTO < 1sec)



Optimal thrust profile



Conclusion

- MColl prototype software result of multi-year R&D effort at JPL
  - Goal: enable low-thrust optimization in MONTE
- The first year spent investigating various collocation and mesh refinement methods
- See paper for more details about algorithm
- Software is easy to use and has been tested on a variety of example problems
- Next step: infusion into MONTE

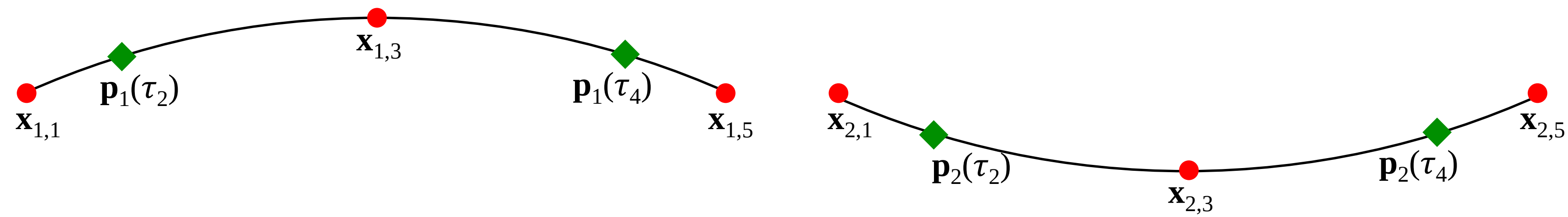
Back-Up Slides

Node Placement Strategies
5th degree polynomials

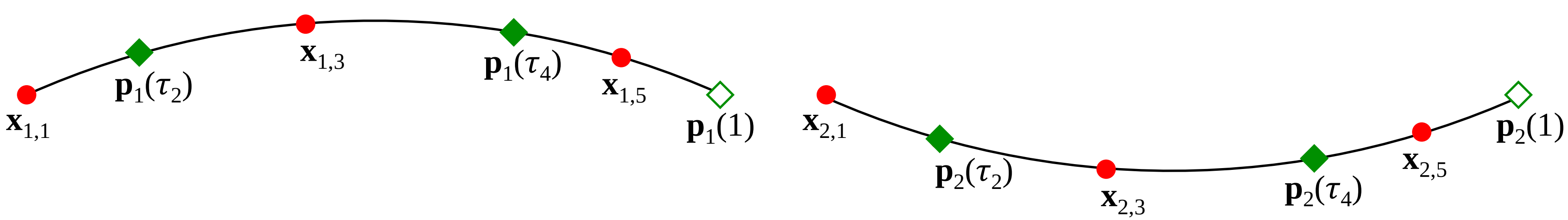
Jet Propulsion Laboratory
California Institute of Technology

◆ Constrained node
● Variable node

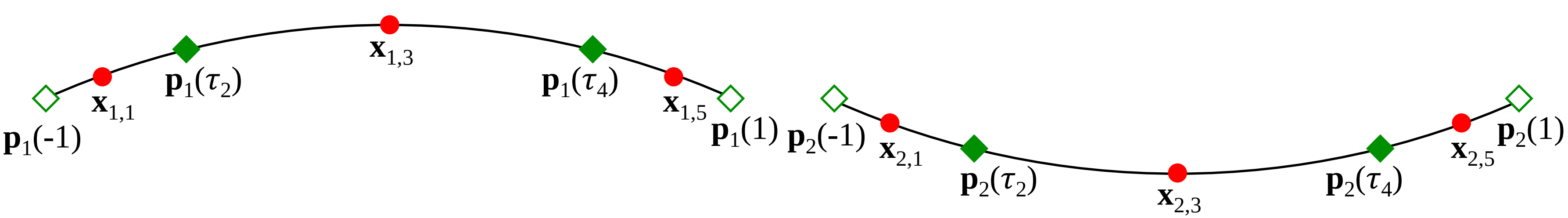
Legendre-Gauss-Lobatto (LGL) Points (8th order method)



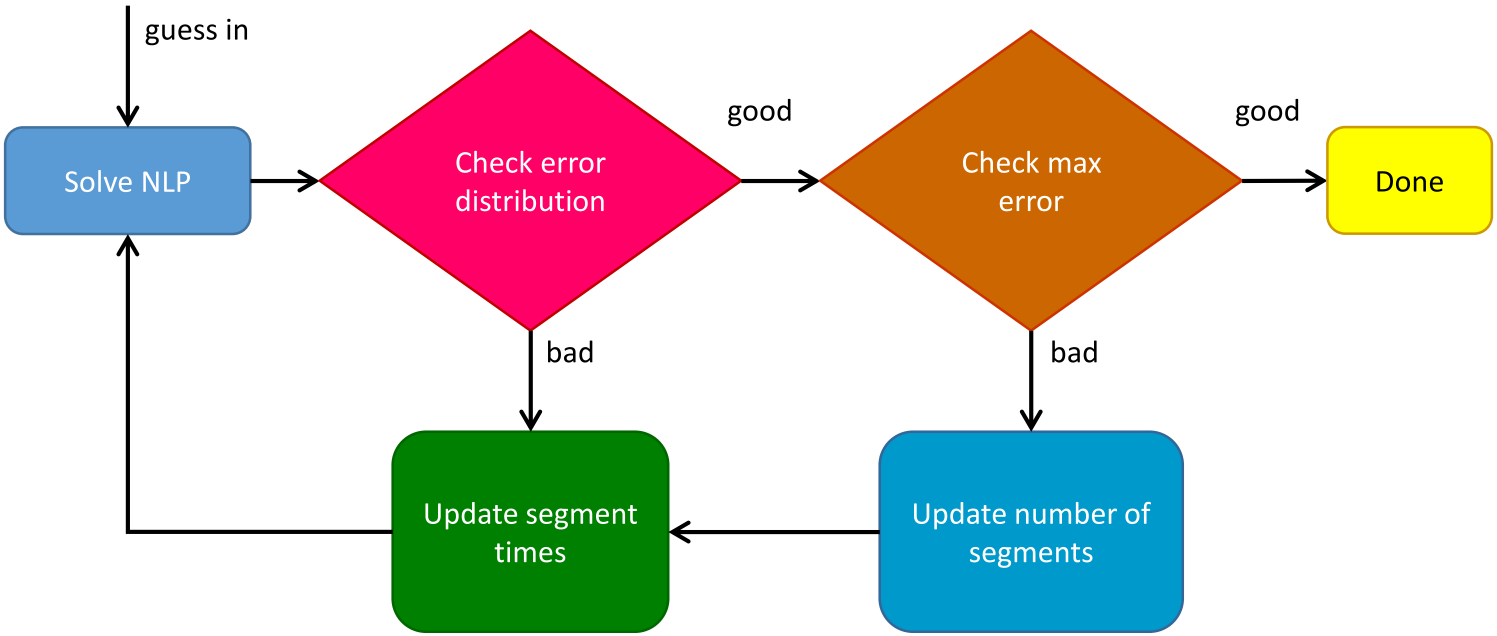
Legendre-Gauss-Radau (LGR) Points (9th order method)



Legendre-Gauss (LG) Points (10th order method)

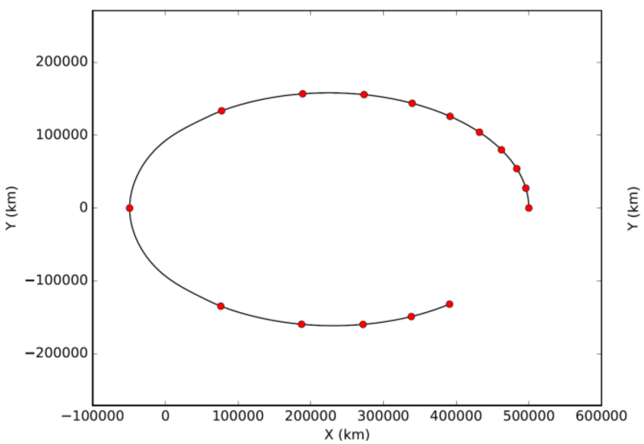


Mesh Refinement Diagram

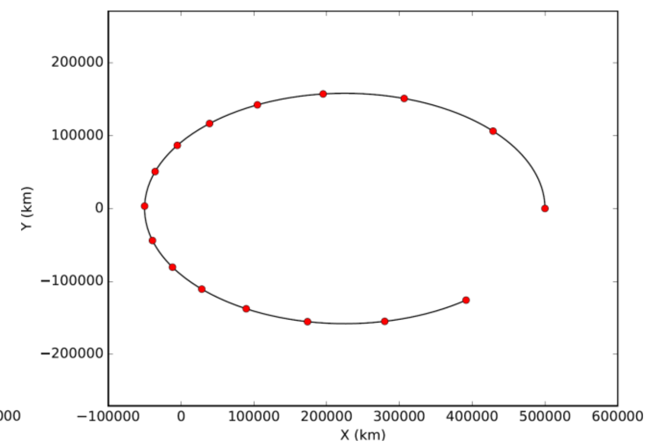


Mesh Refinement Demonstration
Two-Body Ellipse Example

Initial coarse mesh
Segments equally spaced in time



Number segments unchanged
Error equally distributed



Fully refined mesh
Number of segments updated
Error Equally distributed